1. Goal:

void hanoi(int n, char from, char to, char temp);

它的運作模式是從from柱子拿n個盤子放到to柱子上面。

1. Basic case:

if(n==1){

printf("move %d from %c to %c\n",n,from,to);}

1. Inductive case:

else{

hanoi(n-1, from,temp,to);

printf("move %d from %c to %c\n",n,from,to);

hanoi(n-1,temp,to,from);

}

1. Recursion:

void hanoi(int n, char from, char to, char temp){

if(n==1){

printf("move %d from %c to %c\n",n,from,to);

return ;

}

else{

hanoi(n-1, from,temp,to);

printf("move %d from %c to %c\n",n,from,to);

hanoi(n-1,temp,to,from);

}

}

1. Bonus Section: Time Complexity Analysis

n=1;

Hanoi(1)=1

n=2;

Hanoi(2)=2\*Hanoi(1)+1=3

n=3;

Hanoi(3)=2\*Hanoi(2)+1=2\*(2\*Hanoi(1)+1)+1=7

…

…

…

…

n=k;

Hanoi(k)=2\*Hanoi(k-1)+1=2\*(2\*(2\*(…..(2\*(2\*Hanoi(1)+1)+1)…….)+1)+1)+1

=2^(k-1)+2(k-2)+……+2^1+2^0=1(2^k-1)/(2-1)=2^k-1

Ans:2^n-1

Replace Loops with Recursion

#include <stdio.h>

#define MAX 10

#define swap(a, b) if (a != b) {(\*a)^=(\*b); (\*b)^=(\*a); (\*a)^=(\*b);}

void selection\_sort(int\* a, int min, int i,int times);

int a[MAX] = {3, 7, 9, 1, 4, 0, 8, 6, 2, 5};

int main(void) {

selection\_sort(a, 0, 1, 0);

for (int i = 0; i < MAX; i++) {

printf("%d ", a[i]);

}

return 0;

}

void selection\_sort(int \*a, int min, int i, int times){

if(times==MAX-1){

return;

}

if(i==MAX){

swap(&a[min], &a[times]);

times++;

selection\_sort(a,times,times+1,times);

return;

}

if(a[min]>a[i]){

min=i;

i++;

selection\_sort(a,min,i,times);

}

else{

i++;

selection\_sort(a,min,i,times);

}

}

Replace Recursion with Loops

完全沒想法，看答案理解。

#include <stdio.h>

#define MAX 3

typedef struct {

int n, from, to, temp;

} Record;

Record stack[MAX];

int top = -1;

void move\_hanoi(int n, int from, int to, int temp);

int main(void) {

move\_hanoi(MAX, 1, 3, 2);

return 0;

}

void move\_hanoi(int n, int from, int to, int temp) {

stack[++top] = (Record) {n, from, to, temp};

while (top != -1) {

Record record = stack[top--];

if (record.n == 1 || record.temp == -1) {

printf("Move disk %d from %d to %d\n", record.n, record.from, record.to);

continue;

}

stack[++top] = (Record) {record.n - 1, record.temp, record.to, record.from};

stack[++top] = (Record) {record.n, record.from, record.to, -1};

stack[++top] = (Record) {record.n - 1, record.from, record.temp, record.to};

}

}

這次的作業讓我體會到遞迴真的很難且博大精深，儘管我第二題有寫出自己的解法，但其實並沒有依循上述的四步驟，看完解答案才發覺自己沒有考慮到的地方。儘管有些受挫，但也激起我想要把遞迴搞懂的心，期待之後的題目。